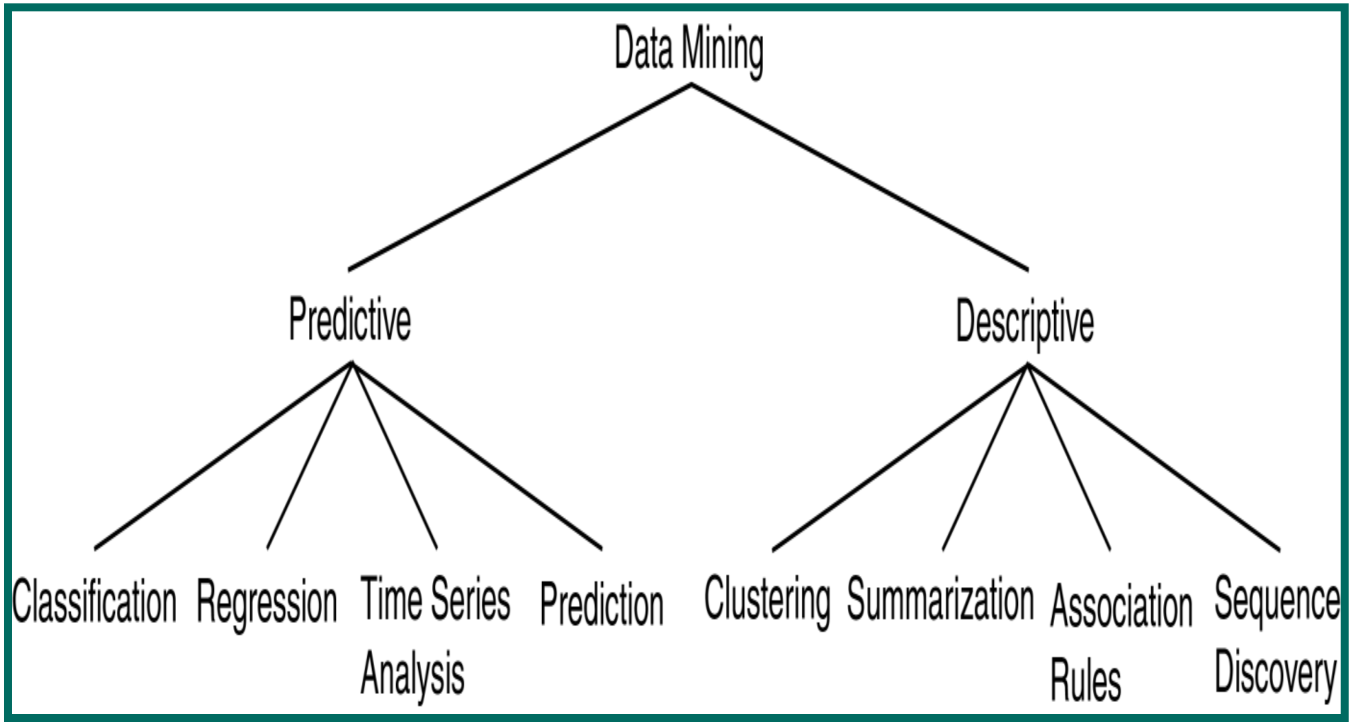
# Data Science

Lecture 3

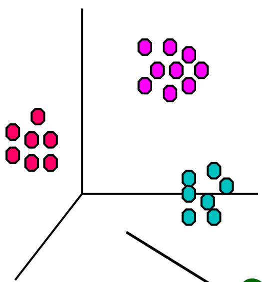
Classification: Basics and Algorithms

Jnaneshwar Bohara

# Data Mining Models and Tasks

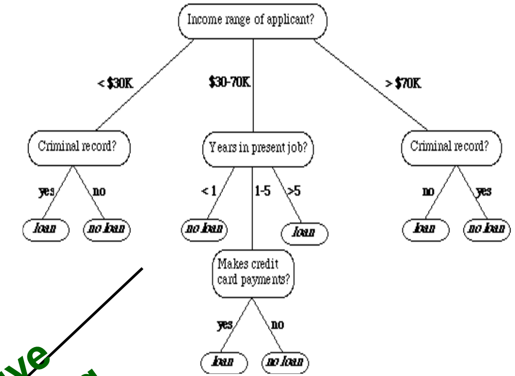


# Data Mining Tasks



Data

Clustering

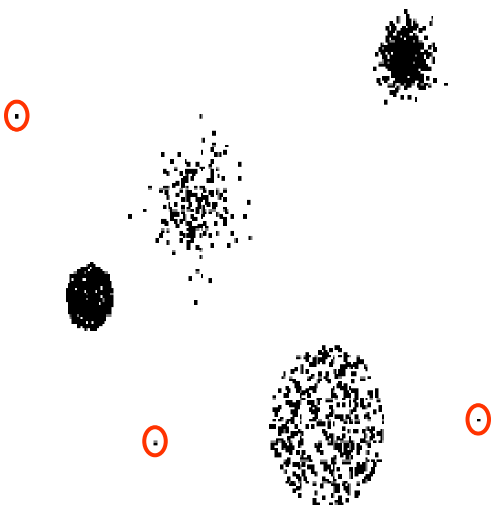



Predictive Modeling

| Tid | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |
| 11 | No | Married | 60K | No |
| 12 | Yes | Divorced | 220K | No |
| 13 | No | Single | 85K | Yes |
| 14 | No | Married | 75K | No |
| 15 | No | Single | 90K | Yes |

Association Rules

Anomaly Detection



# Basic Data Mining Tasks

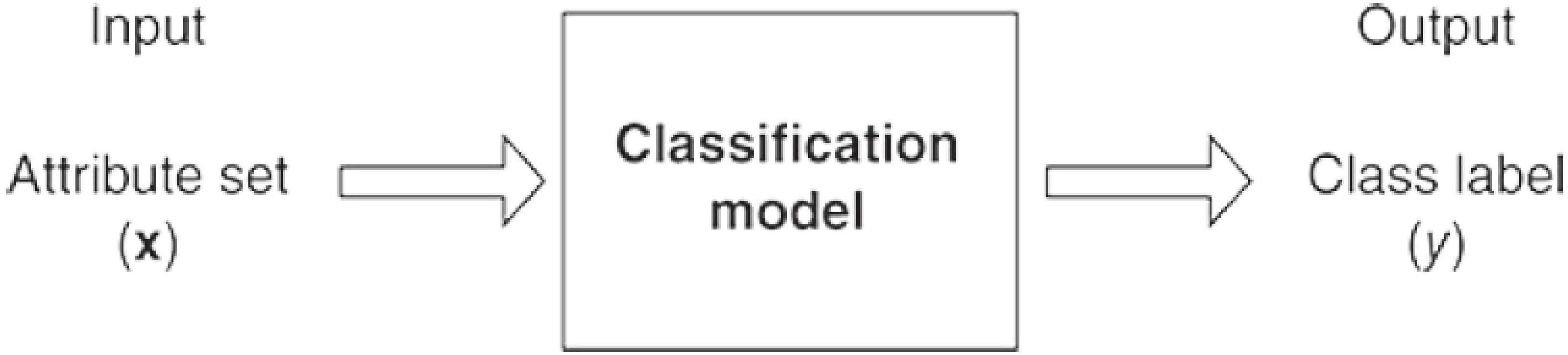
- ***Classification*** maps data into predefined groups or classes
  - Supervised learning
  - Pattern recognition
  - Prediction
- ***Regression*** is used to map a data item to a real valued prediction variable.
- ***Clustering*** groups similar data together into clusters.
  - Unsupervised learning
  - Segmentation
  - Partitioning

# Classification: Definition

- Given a collection of records (training set )
  - Each record is characterized by a tuple ($\boldsymbol{x}$,$y$), where $\boldsymbol{x}$ is the attribute set and $y$ is the class label
    - $\boldsymbol{x}$: attribute, predictor, independent variable, input
    - $y$: class, response, dependent variable, output

- Task:
  - Learn a model that maps each attribute set $\boldsymbol{x}$ into one of the predefined class labels $y$

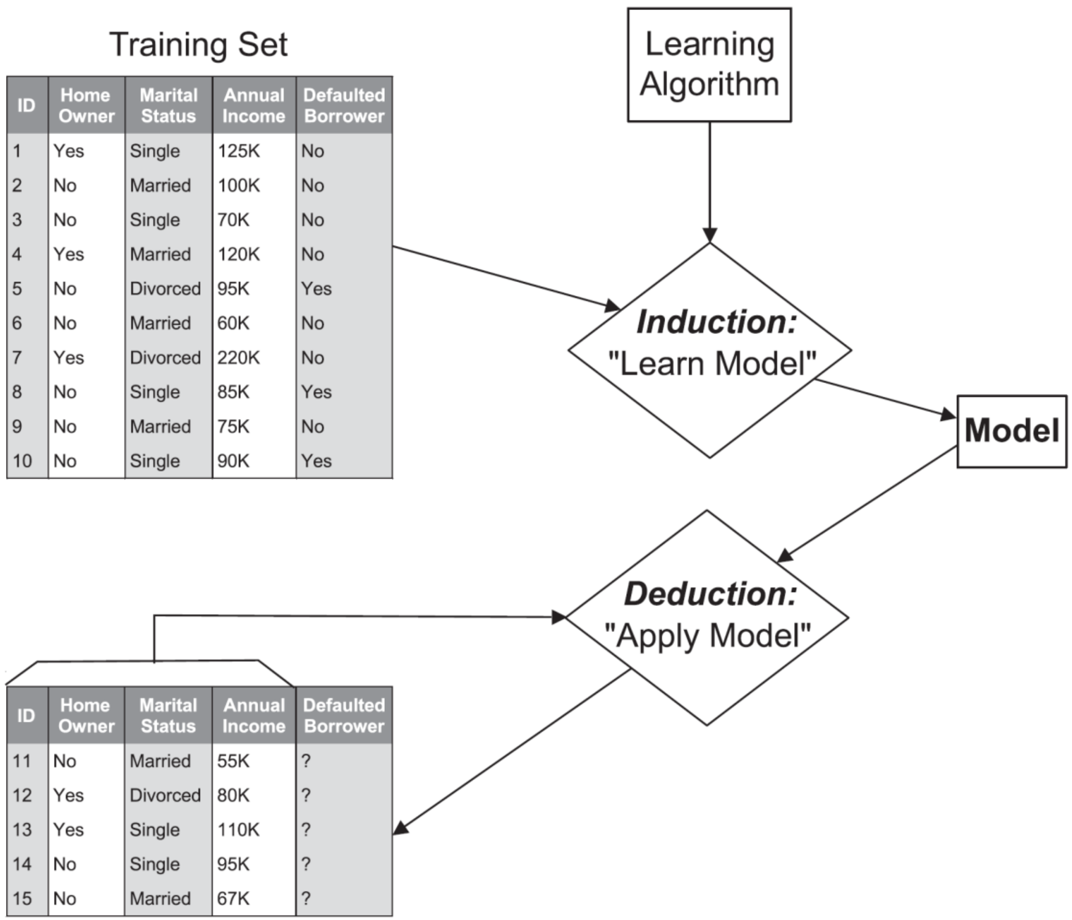
# Classification Task



# Examples of Classification Task

| Task | Attribute set, $x$ | Class label, $y$ |
|---|---|---|
| Categorizing email messages | Features extracted from email message header and content | spam or non-spam |
| Identifying tumor cells | Features extracted from x-rays or MRI scans | malignant or benign cells |
| Cataloging galaxies | Features extracted from telescope images | Elliptical, spiral, or irregular-shaped galaxies |

# General Framework for Classification




Training Set

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 11 | No | Married | 55K | ? |
| 12 | Yes | Divorced | 80K | ? |
| 13 | Yes | Single | 110K | ? |
| 14 | No | Single | 95K | ? |
| 15 | No | Married | 67K | ? |

Test Set

**Figure 3.3.** General framework for building a classification model.

# General Framework for Classification

- Classification is the task of assigning labels to unlabeled data instances and a **classifier** is used to perform such a task
- A classifier is typically described in terms of a model
- The model is created using a given a set of instances, known as the **training set,** which contains attribute values as well as class labels for each instance
- The systematic approach for learning a classification model given a training set is known as a **learning algorithm**

# General Framework for Classification

- The process of using a learning algorithm to build a classification model from the training data is known as **induction**
- This process is also often described as “learning a model” or “building a model.”
- This process of applying a classification model on unseen test instances to predict their class labels is known as **deduction**
- Thus, the process of classification involves two steps: applying a learning algorithm to training data to learn a model, and then applying the model to assign labels to unlabeled instances

# Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
  - Neural Networks, Deep Neural Nets

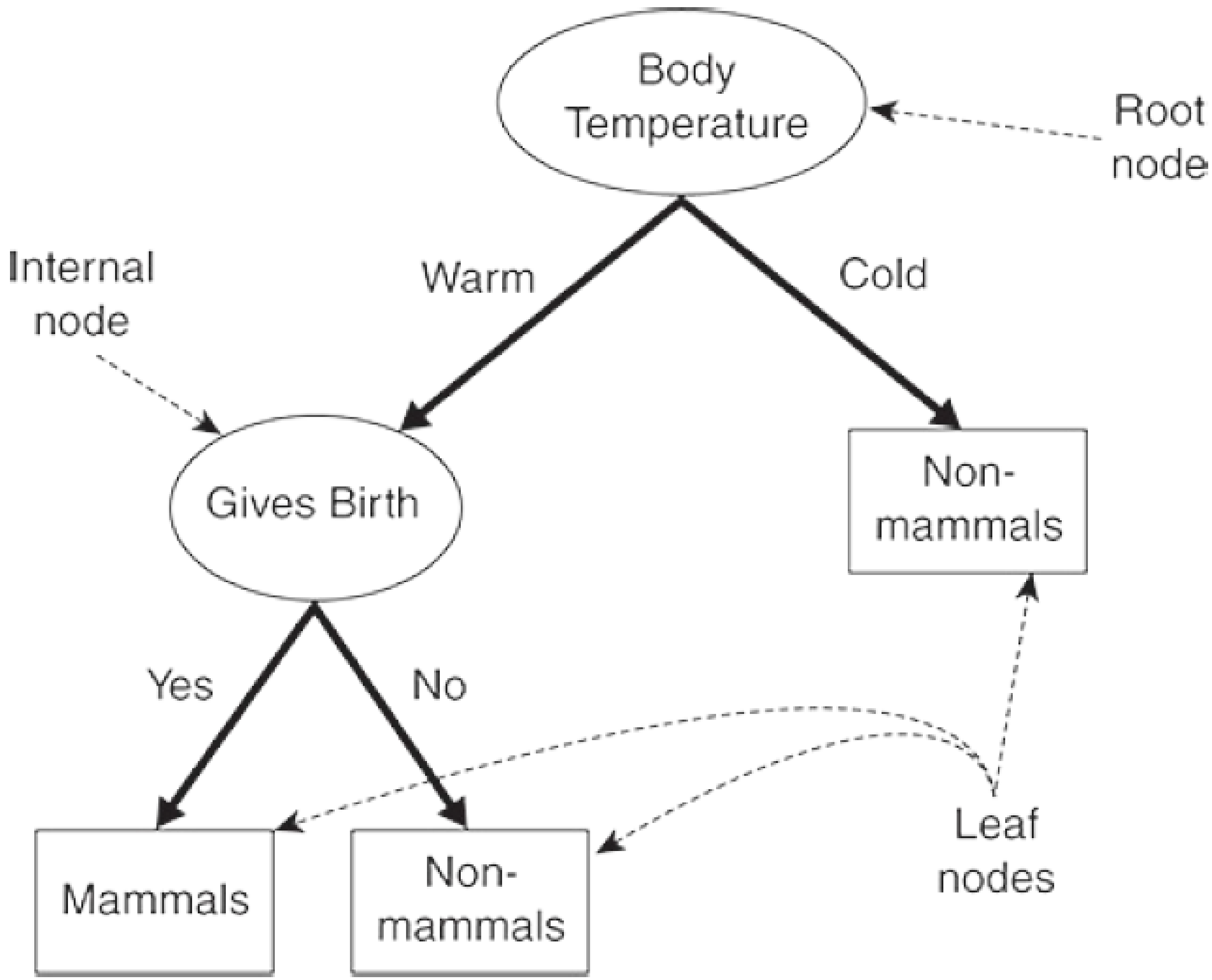
# Decision Tree classifier

- A decision tree is tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a classification or decision.
- Decision tree is a classifier in the form of a tree structure where a **leaf node** indicates the class of instances, a **decision node** specifies some test to be carried out on a single attribute value with one branch and sub-tree for each possible outcome of the test.
- A decision tree can be used to classify an instance by starting at root of the tree and moving through it until leaf node. The leaf node provides the corresponding class of instance.

# Nodes in Decision Tree

- A **root node**, with no incoming links and zero or more outgoing links
- **Internal nodes**, each of which has exactly one incoming link and two or more outgoing links
- **Leaf** or **terminal** nodes, each of which has exactly one incoming link and no outgoing links
- Every leaf node in the decision tree is associated with a class label
- The **nonterminal** nodes, which include the root and internal nodes, contain **attribute test conditions** that are typically defined using a single attribute
- Each possible outcome of the attribute test condition is associated with exactly one child of this node

# Nodes in Decision Tree

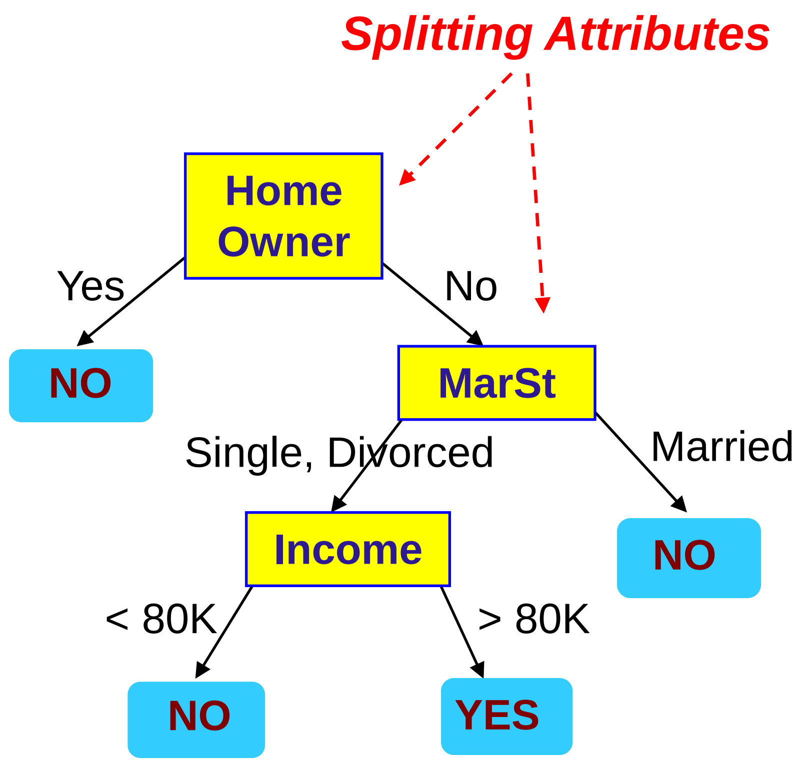


# Example of a Decision Tree

categorical categorical continuous class

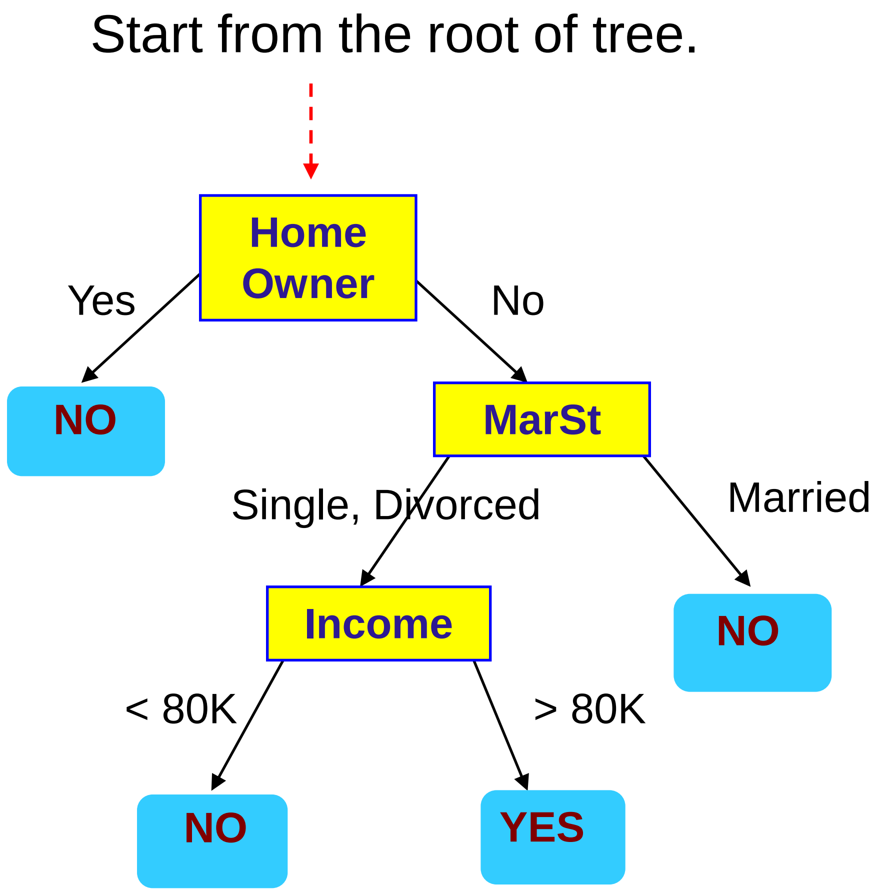
| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

**Training Data**




**Model: Decision Tree**

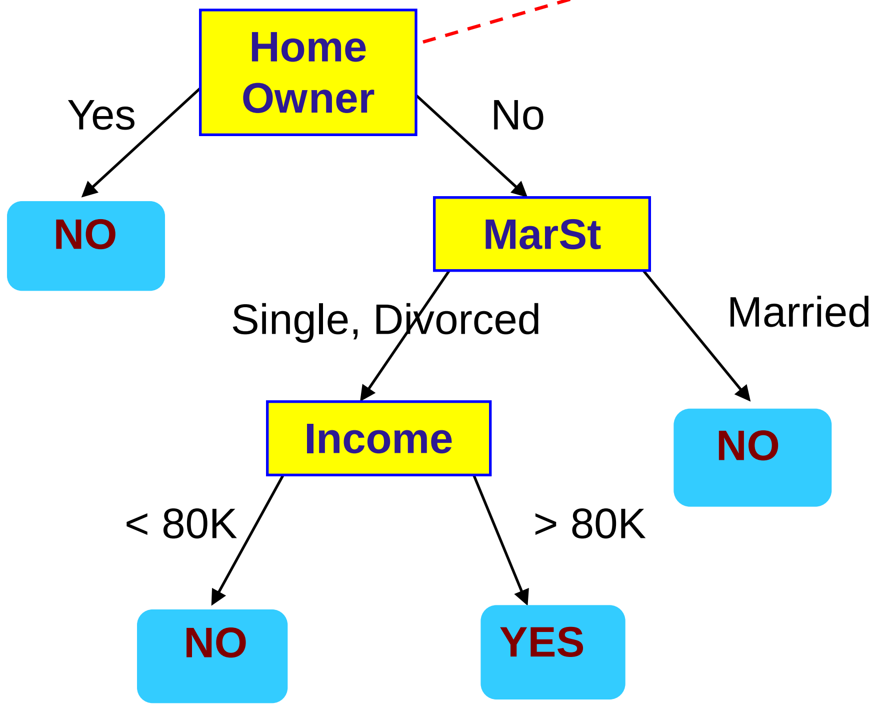
# Apply Model to Test Data




## Test Data

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

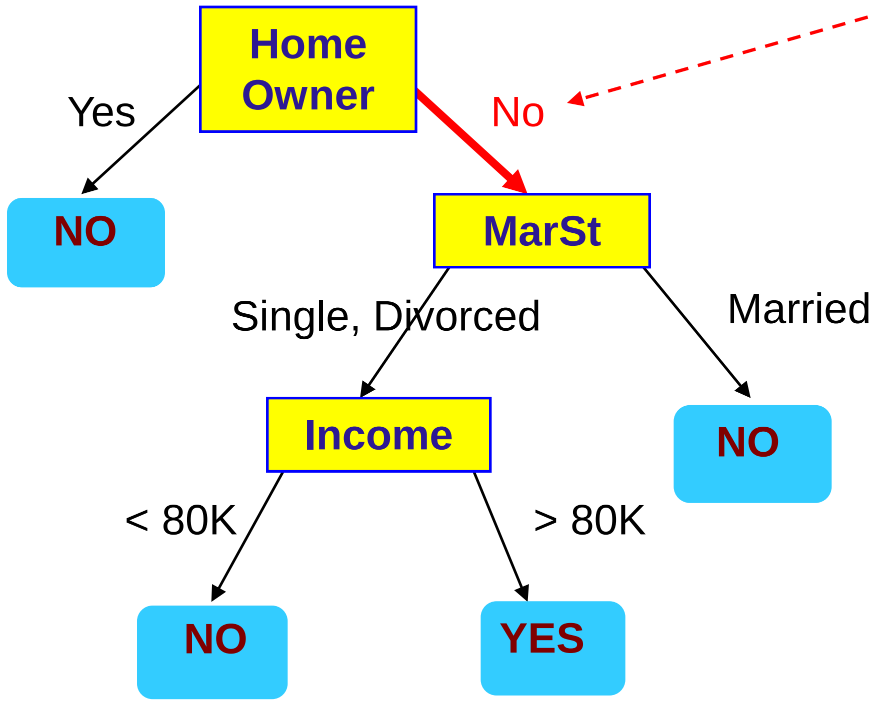
# Apply Model to Test Data




## Test Data

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

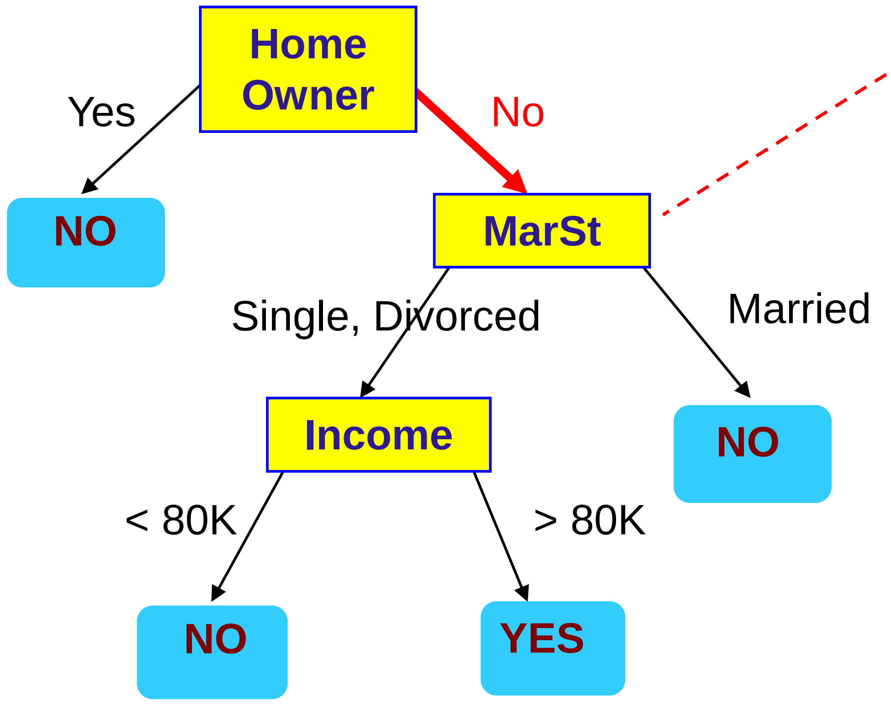
# Apply Model to Test Data




## Test Data

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

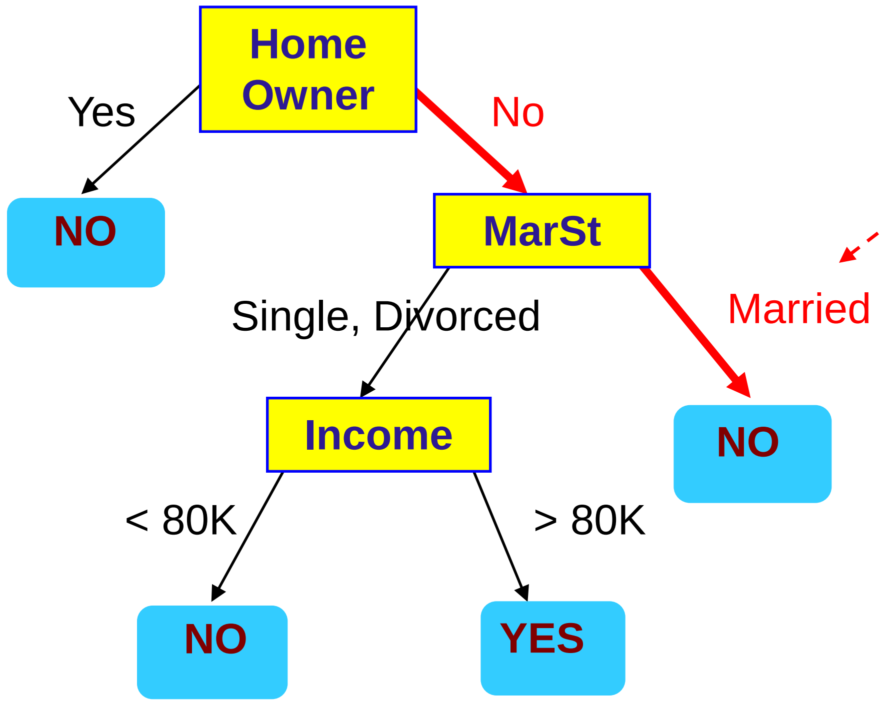
# Apply Model to Test Data




## Test Data

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

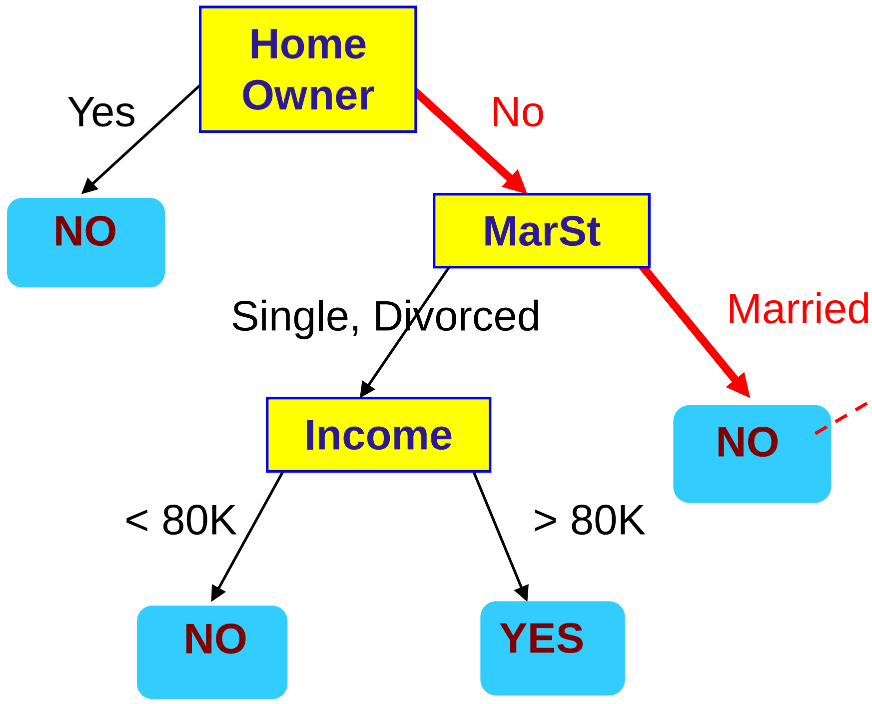
# Apply Model to Test Data




**Test Data**

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

# Apply Model to Test Data




**Test Data**

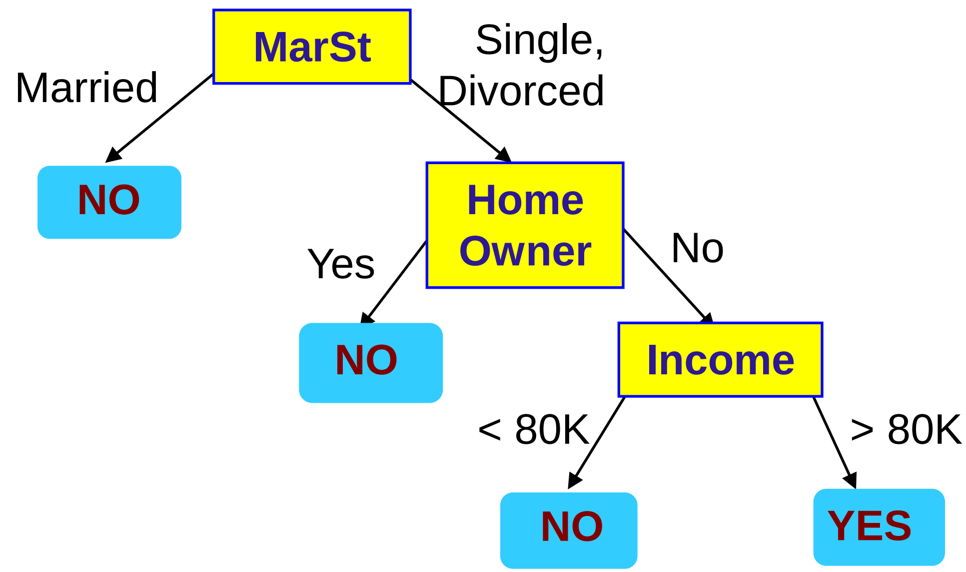
| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | ? |

Assign Defaulted to "No"

# Another Example of Decision Tree

categorical categorical continuous class

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

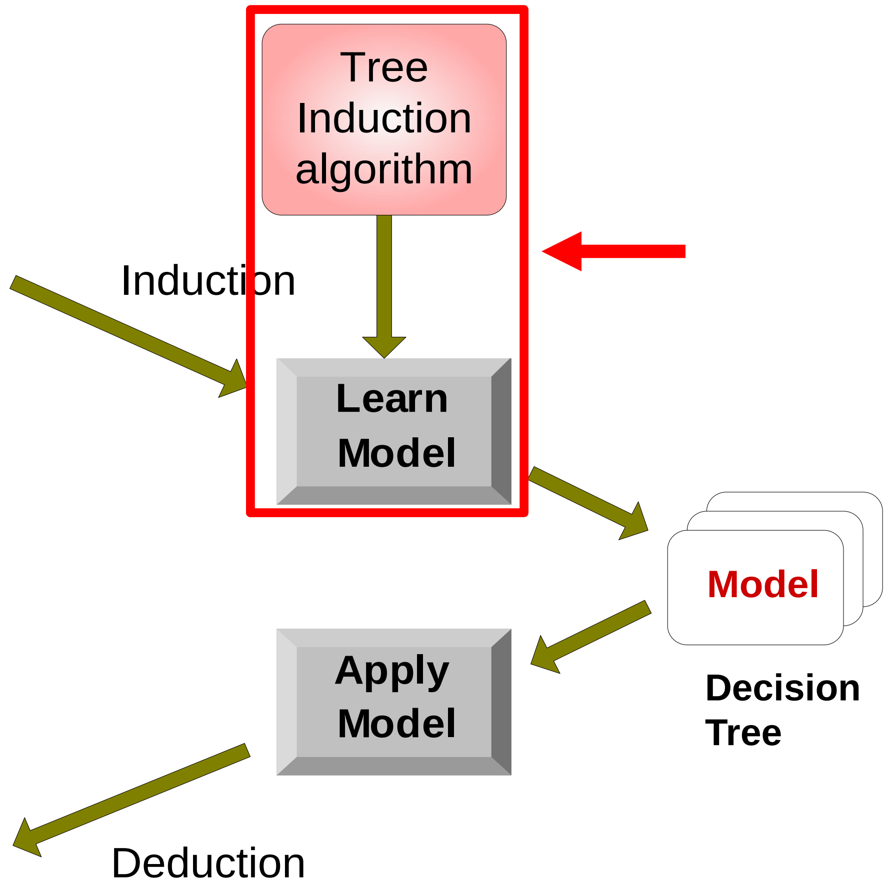



**There could be more than one tree that fits the same data!**

# Decision Tree Classification Task

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set




| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set

# Decision Tree Induction

- Many Algorithms:
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

# Hunt's Algorithm

- In Hunt's algorithm, a decision tree is grown in a recursive fashion
- The tree initially contains a single root node that is associated with all the training instances
- If a node is associated with instances from more than one class, it is expanded using an attribute test condition that is determined using a **splitting criterion**
- A child leaf node is created for each outcome of the attribute test condition and the instances associated with the parent node are distributed to the children based on the test outcomes

# Hunt’s Algorithm

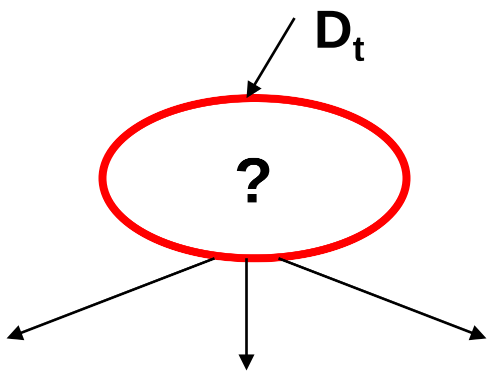
- This node expansion step can then be recursively applied to each child node, as long as it has labels of more than one class
- If all the instances associated with a leaf node have identical class labels, then the node is not expanded any further
- Each leaf node is assigned a class label that occurs most frequently in the training instances associated with the node

# General Structure of Hunt's Algorithm

- Let $D_t$ be the set of training records that reach a node t

- General Procedure:
  - If $D_t$ contains records that belong the same class $y_t$, then t is a leaf node labeled as $y_t$
  - If $D_t$ contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |



# Hunt's Algorithm

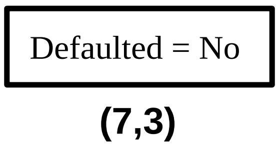
Defaulted = No

**(7,3)**

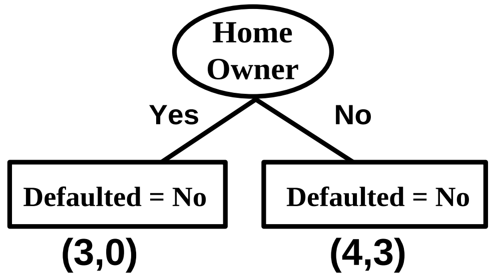
(a)

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | **No** |
| 2 | No | Married | 100K | **No** |
| 3 | No | Single | 70K | **No** |
| 4 | Yes | Married | 120K | **No** |
| 5 | No | Divorced | 95K | **Yes** |
| 6 | No | Married | 60K | **No** |
| 7 | Yes | Divorced | 220K | **No** |
| 8 | No | Single | 85K | **Yes** |
| 9 | No | Married | 75K | **No** |
| 10 | No | Single | 90K | **Yes** |

# Hunt’s Algorithm



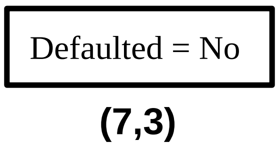

(a)



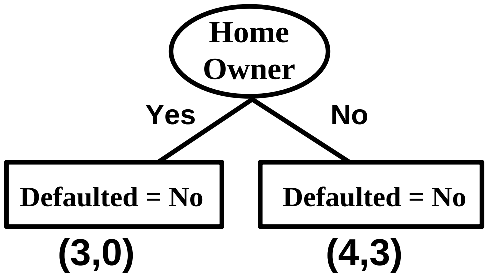

(b)

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

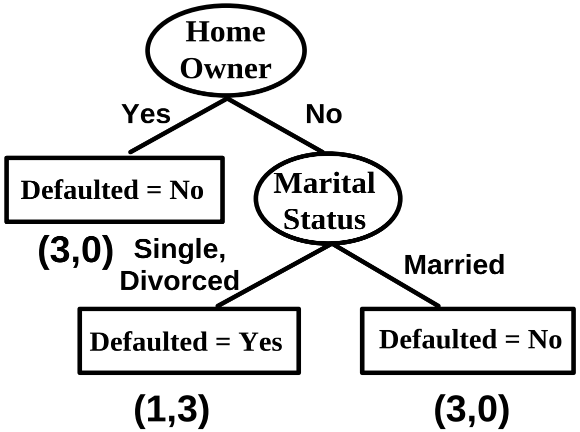
# Hunt's Algorithm




(a)



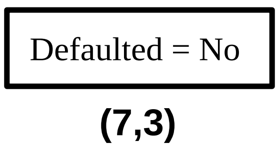

(b)



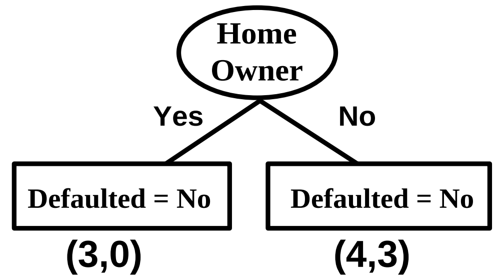

(c)

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

# Hunt's Algorithm

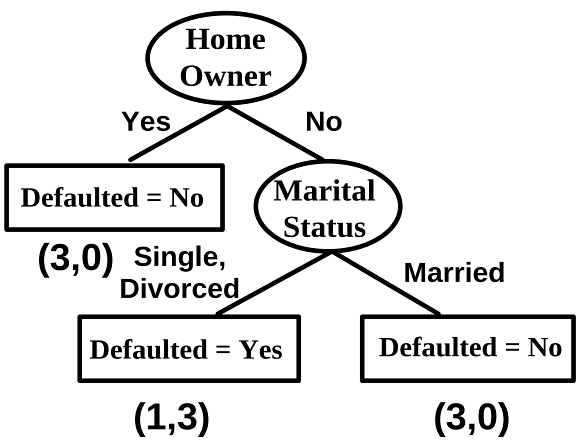



(a)

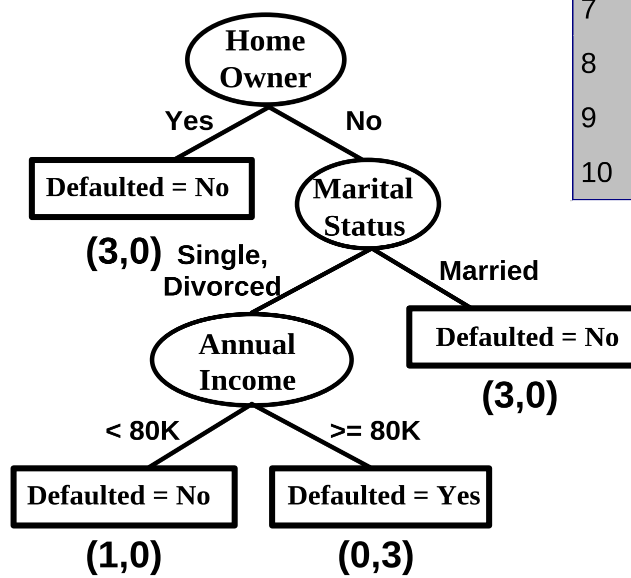



(b)

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |




(c)




(d)

# Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for expressing test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

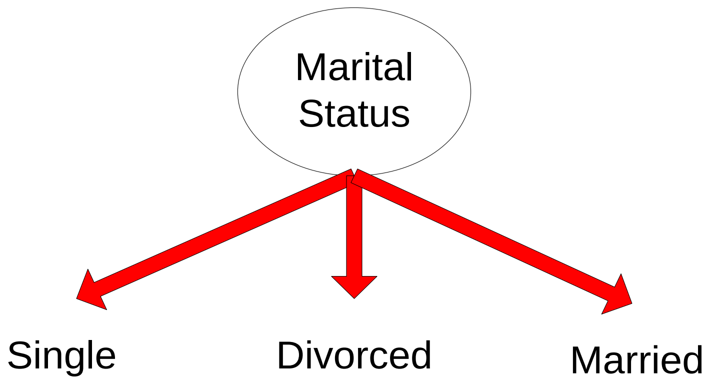
- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

# Methods for Expressing Test Conditions

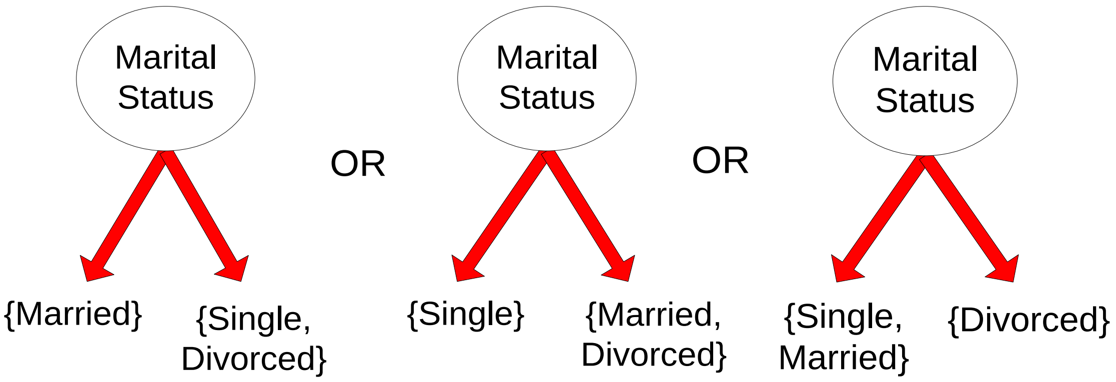
- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

# Test Condition for Nominal Attributes

- Multi-way split:
  – Use as many partitions as distinct values.

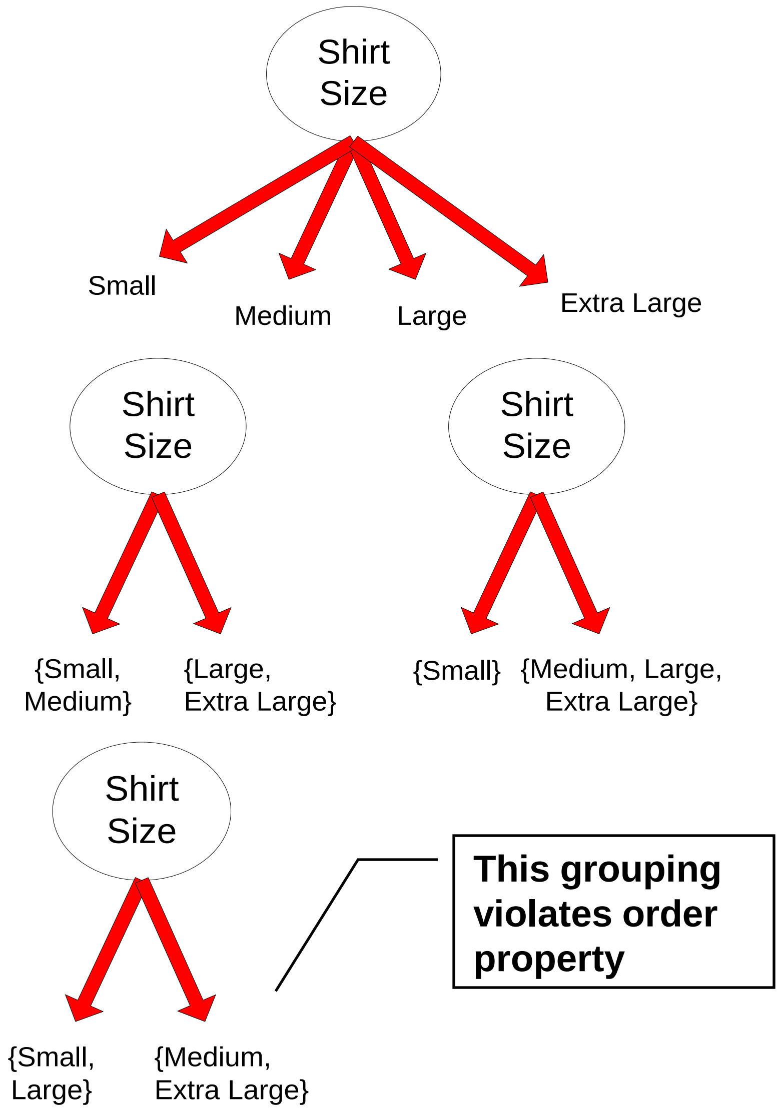



- Binary split:
  – Divides values into two subsets

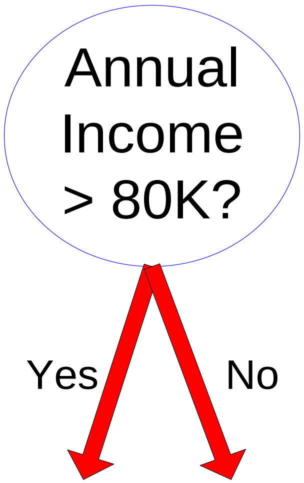


# Test Condition for Ordinal Attributes

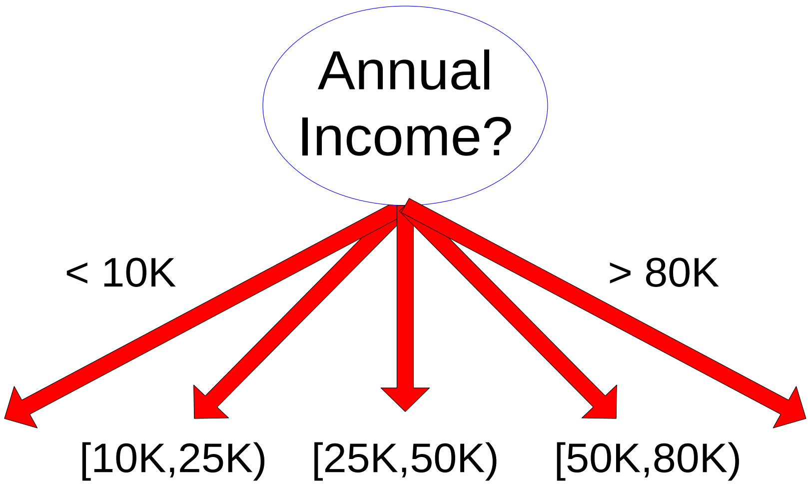
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



# Test Condition for Continuous Attributes




(i) Binary split




(ii) Multi-way split

# Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute

    Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

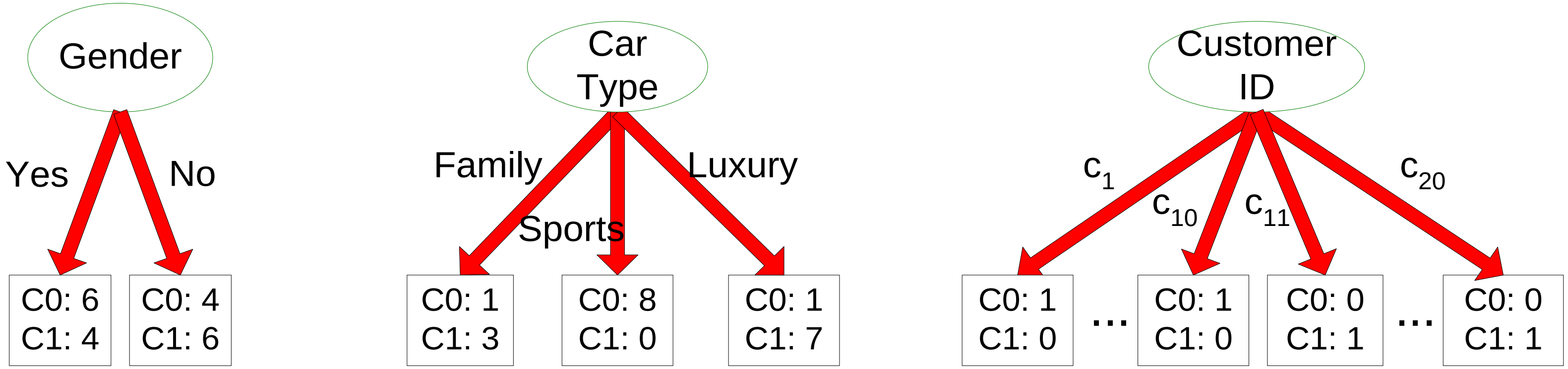
    - Static – discretize once at the beginning
    - Dynamic – repeat at each node

  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

# How to determine the Best Split

| Customer Id | Gender | Car Type | Shirt Size | Class |
|---|---|---|---|---|
| 1 | M | Family | Small | C0 |
| 2 | M | Sports | Medium | C0 |
| 3 | M | Sports | Medium | C0 |
| 4 | M | Sports | Large | C0 |
| 5 | M | Sports | Extra Large | C0 |
| 6 | M | Sports | Extra Large | C0 |
| 7 | F | Sports | Small | C0 |
| 8 | F | Sports | Small | C0 |
| 9 | F | Sports | Medium | C0 |
| 10 | F | Luxury | Large | C0 |
| 11 | M | Family | Large | C1 |
| 12 | M | Family | Extra Large | C1 |
| 13 | M | Family | Medium | C1 |
| 14 | M | Luxury | Extra Large | C1 |
| 15 | F | Luxury | Small | C1 |
| 16 | F | Luxury | Small | C1 |
| 17 | F | Luxury | Medium | C1 |
| 18 | F | Luxury | Medium | C1 |
| 19 | F | Luxury | Medium | C1 |
| 20 | F | Luxury | Large | C1 |

**Before Splitting: 10 records of class 0,**
**10 records of class 1**




**Which test condition is the best?**

# How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

| C0: 5 | C0: 9 |
|---|---|
| C1: 5 | C1: 1 |
| **High degree of impurity** | **Low degree of impurity** |

# Measures of Node Impurity

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node **t**, and $\boldsymbol{c}$ is the total number of classes

- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

# Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

**Gain = P - M**

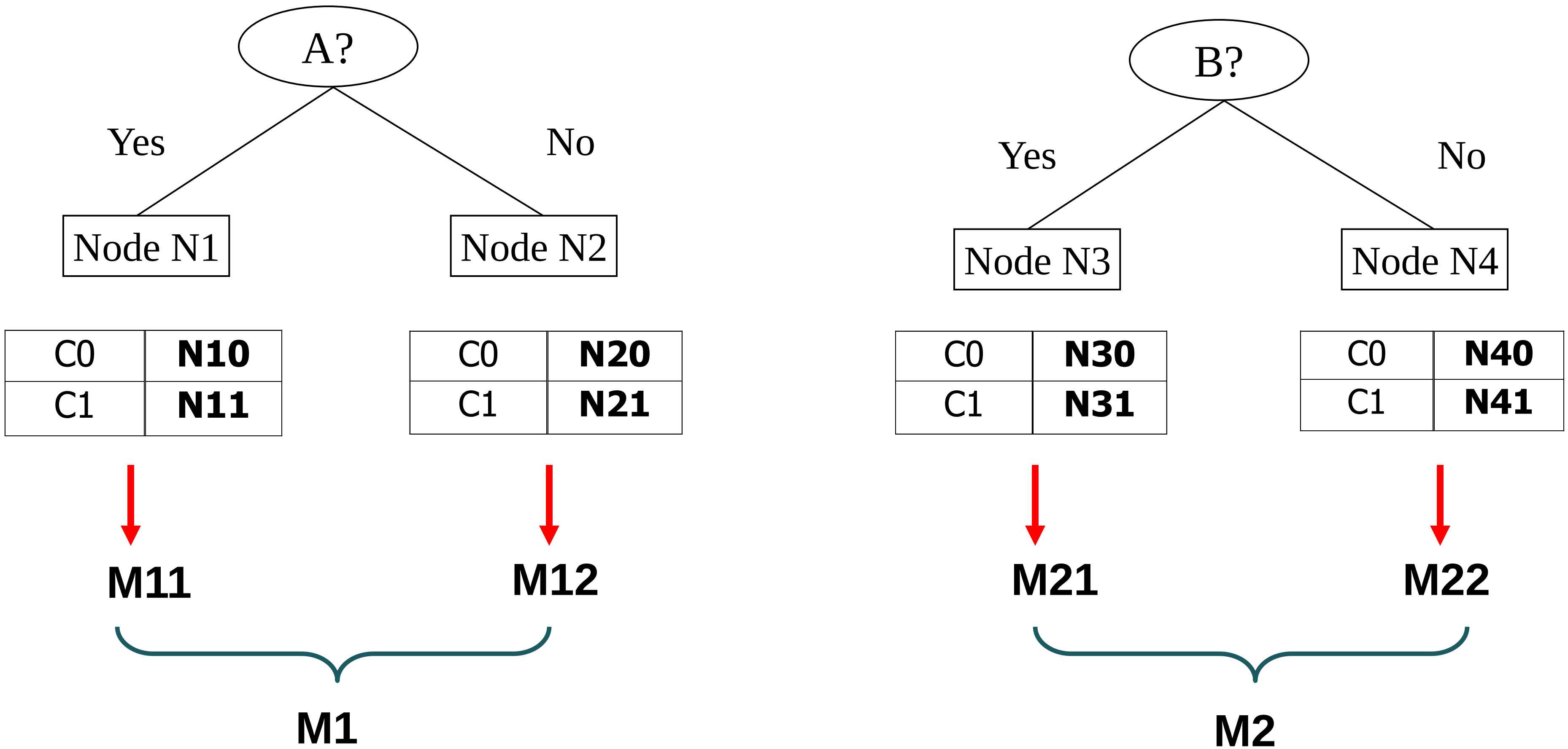
or equivalently, lowest impurity measure after splitting (M)

# Finding the Best Split

**Before Splitting:**

| | |
|---|---|
| C0 | **N00** |
| C1 | **N01** |

→ **P**




**Gain = P – M1 vs P – M2**

# Decision Tree Based Classification

- Advantages:
  - Relatively inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant attributes
  - Can easily handle irrelevant attributes (unless the attributes are interacting)
- Disadvantages: .
  - Due to the greedy nature of splitting criterion, interacting attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.
  - Each decision boundary involves only a single attribute